DELGEN

X-Graph Software Module

xgFAT Users Manual

Version 1.0-Beta – 14th August 2007

Printed in Luxembourg.
Document Nr: XGFAT-001

Content

Figures

Tables

1 Welcome

1.1 Introduction

This X-Graph software module adds a full 'embedded' FAT12-FAT16-FAT32 compatible filesystem to your X-Graph projects. And it uses a minimum memory footprint.

It's compatible with all X-Graph mass storage devices and has an open HAL structure to add custom mass storage devices.

IMPORTANT: This X-Graph software module is currently under development. The information in this manual is provided as-is and might be changed by DELGEN at any time. Contact DELGEN for a release date of the final library.

1.2 How This Book Is Organized

You can find following chapters in it:

Chapter 1 contains a view on all the information in this book.

Chapter 2 includes all info on the xgFAT X-Graph software module.

1.3 More Questions

If you have questions while using this X-Graph software module, check first if the information is available in this book. If you cannot find the answer check the information and forum on the X-graph website (www.x-graph.be). Finally you can also contact your local distributor or the X-Graph technical support by e-mail (techsup@x-graph.be).

This manual includes all available documentation on this X-Graph software module. It is strongly advised to download and read documentation on the Rabbit processor available from the Rabbit Semiconductor (www.rabbitsemiconductor.com) website. This manual is complimentary to the documentation found on these websites.

2 X-Graph FAT library (xgFAT)

A complete FAT12-FAT16-FAT32 library which supports all X-Graph mass storage devices and offers an open cache and function structure optimized for embedded X-Graph designs.
The library uses an application dependent caching algorithm with delayed writes. It supports simultaneous use of all available mass storage devices.

xgFAT reads and writes to mass storage devices fully compliant to the Microsoft FAT specification. All data can be read and written with a standard FAT compatible PC (Mac, Linux, Windows, ...). Long file names are preserved.

The X-Graph software license does not allow you to use this library with non-X-Graph hardware. The xgFAT libraries have a locking mechanism which prevents the libs from being used on not X-Graph hardware.

2.1 FAT Standard

2.1.1 Specification Compliance

The xgFAT library is written in accordance with the Microsoft Hardware White Paper V1.03, Dec 6, 2000 ‘ Microsoft Extensible Firmware Initiative FAT32 File System Specification’.

xgFAT supports mass storage devices formatted with FAT12, FAT16 and FAT32.
Full long file name support (LFN) will be available in the next release. Currently long file name entries are preserved.

2.1.2 Path Length

Many sources say that paths are limited to 80 or 128 characters, but these limitations are due to DOS peculiarities, but not the FAT format. With a FAT file system, one can have infinitely long paths. The general recommended maximum length of a path is 256 characters, including terminating null character. Long names are up to 256 characters long, including extension and the terminating null character. This limitation is artificial, and the long names on the disk can actually be longer than 256 characters. Again, some limitations were created by the software that serves FAT (Microsoft).

Above mentioned Microsoft document states the following on this subject:
The total path length of a short name cannot exceed 80 characters (64 char path + 3 drive letter + 12 for 8.3 names + NUL) including the trailing NUL.
The total path length of a long name can not exceed 260 characters, including the trailing NUL.

The xgFAT library limits the path length to 256 characters. This maximum length of a directory path, includes the drive letter, column, leading slash, trailing slash, null terminator, filename and extension.
This should be an acceptable rule for embedded applications. If longer paths are detected, these are just skipped by the xgFAT library.

2.1.3 Time/Date Limit

The DynamicC clock is limited from 1980 up to 2047, thus FAT dates created by xgFAT have the same limit.

2.1.4 Optional DIR_LstAccDate

The ‘Directory Last Access Date’ is not written by xgFAT, to speed up things. Writing this field is optional (10.1.1).

2.1.5 Sector Size

Only FAT volumes with a 512 byte sector size are supported. As all modern hardware only supports this sector size, this shouldn’t be a problem.

2.1.6 Multiple FATs

FAT volumes normally use two identical FAT tables. This is done to provide redundancy so that if a sector goes bad in one of the FATs, that data is not lost because it’s duplicated in the other FAT. On non-disk-based media (SDCard, Flash, …) and on modern hard discs (which automatically handle defective sectors), this redundancy is a useless feature. A single FAT will do the job.

xgFAT uses a single FAT for on-board Flash, battery-back SRAM and Z-World Serial Flash volumes.

xgFAT only reads the first FAT table copy (there can be up to 255 copies). Errors while reading this first copy will result in a fatal error.
For FAT writes, all FAT copies are written, always.

2.2 xgFAT Structure

2.2.1 Abstraction Layers

The xgFAT libraries use three abstraction layers.

xgFS.LIB (layer 1) includes the xgFAT interface to the UI. The lib contains a set of ‘C’-style file access functions.

The xgFAT kernel (layer 2) includes 4 libraries. FS_FAT.LIB has all the FAT handling related functions. Partition detection and volume mounting is handled in FS_VOLUME.LIB. A file stack is maintained in FS_FILE.LIB. These 3 files are X-Graph copy protected. FS_CACHE.LIB arranges actual reads and writes, and the caches. Several versions of the FS_CACHE library are available with different kinds of caching algorithms.

The HAL (layer 3) includes several mass storage device drivers (MSDD). DELGEN offers MSDD’s for the MMC/SDCard socket, CF slots, on-board Flash and battery-backup SRAM, and Z-Worlds Serial-Flash. A MSDD for IDE hard discs will be available soon. Users can write MSDD’s for any other MSD attached to the X-Graph.

2.2.2 User Interface

To include xgFAT support, only the xgFS.LIB library needs to be ‘use’d’. All other xgFAT files are automatically included.

Alternatively the xgFS_AutoMount.LIB library can also be used. This library will automatically mount all available xgFAT compatible mass storage devices and use xgFS.LIB. In systems where automounting is not desirable this library should not be #use'd.

2.3 FAT Partitions & Nomenclature

Nomenclature:

- Device: a physical mass storage device (i.e. SDCard, Flash, …)
- Partition: a device partition
- Volume: a FAT volume, typically stored in a partition on a device.
  A volume requires a volume identifier (i.e.: C:\, D:\, SDCard:\ …)

2.3.1 Partitions

A word about partitions, especially DOS partitions (also ALL Windows partitions)

- The MBR (sector 0 of the device) contains maximum 4 partitions
- DOS allows 1 (and only 1) ‘primary’ DOS partition in the MBR
- DOS allows 1 (and only 1) ‘DOS extended partition’ in the MBR. Note that the other partitions can be used for Linux, …
- The DOS Extended partition is a container which holds ALL the other DOS ‘partitions’ or logical drives or volumes. Within this partition, the logical drives are stored in a linked structure.
- The DEP has a dummy MBR or ‘extended partition table’ containing 2 partitions which describe a logical drive and a link to the next ‘extended partition table’.
- The number of partitions is limited by the maximum of letters in the alphabet, as DOS volume identifiers are single letters. A and B are reserved which results in a maximum of 24 DOS partitions (on all attached devices).

This xgFAT release only supports primary DOS partitions. Extended partitions are detected and reported, but can not be mounted. For embedded applications this is normally no problem. SDCards and CF cards generally only use a primary partition.

2.3.2 Devices without a Partition Table

Some devices might not have a partition table. As there is no partition table ‘standard’, detecting the partition table presence is not standardized. The xgFAT libs use the jump code detection algorithm. Before checking partition table entries, each device is scanned for a x86 jump instruction in the first 3 bytes of the sector 0. If such a jump is detected, it’s presumed the device does not contain a partition table, but only a DOS MBR. When the MBR security checks fail, a scan for a partition table is done anyhow.

2.3.3 LBA and CHS

The xgFAT library only supports LBA, not the older CHS addressing. For all modern devices (all SDCards for example) this is no problem. DELGEN decided to remove CHS support due to the reduced memory requirements and the lack of real-life test devices. All devices we could find, support LBA.

2.3.4 xgFAT Volume Identifiers

xgFAT volume identifiers can be freely chosen. You’re not limited to a single letter. Any name, i.e. SDCard:\ or CF2:\ or hd0:\ or SerialFlash:\ can be used.

A full filename is defined as:
<volume identifier:><\><directory 1><\directory 2>....<\filename><.extension>
Only '\' is accepted, do not use '/'.

2.4 xgFAT Speed, Memory and Caching

During the development of the xgFAT libraries DELGEN imposed the following targets:

- Use a minimum of root and extended data RAM
- Include an embedded X-Graph optimized function list
- Maximum read/write speed
- Minimum use for program memory (no root memory, only extended)

2.4.1 Embedded FAT Requirements

An X-Graph embedded application can not be compared with a PC. The requirements a PC has for its filesystem are totally different.

What are exactly the requirements of the filesystem for an embedded X-Graph application?

As an X-Graph system only has a limited amount of Flash and RAM memory, it will generally not create a lot of files. Most applications will use the filesystem to read files or sections of files to the data memory, basically using the filesystem to enlarge the available data memory. You could compare the filesystem with a very large extended memory array.

Also the files stored in the filesystem will be written, in most cases, in the factory. End-users are normally not aware of the embedded filesystem. They just use the system with extra features only possible due to the enlarged memory.

Such 'application-read-only' files can be stored on the mass storage device by a PC. Or the DynamicC embedded FTP server, with read/write access to the xgFAT volumes, can be used. This offers the developer the possibility of reading, adding, deleting, etc... files on the xgFAT mass storage devices. In some applications the end-users might also use the FTP server to access for example log files.

Files are used to serve the following list of read-only applications.

- The embedded HTTP server can use webpages stored in the filesystem
- Flash (program memory) upgrades or changes.
- Font files which can be used as direct font buffers.
- Graphic files which can be copied directly to the video sram.
- Database records.
- xgBASIC interpreter program files.
- Help text files to be printed in a help window or console.

The list of 'write-to-file' applications is generally limited to:

- Write log files which are typically text files. The system will most often never read these files. Log files can be accessed with an FTP server and are read/analyzed by service personnel.
- Write screendumps = bitmap files. This feature is normally used by the manual writers to get nice-looking manuals. It's not really an end-user application. Also these files can be easily accessed with an FTP server.
- Write configuration data. But in most cases the amount of data is limited and can be store easily in the X-Graph non-volatile memory arrays.
- Write database records. This seems to be the only real-life end-user write application. Databases stored in the filesystem are useful once they don't fit the on-board flash memory anymore.

2.4.2 Speed & Memory

Let's discuss our targets maximum speed and minimum data memory usage.

The main slow-down issue with existing filesystems is the caching and/or multi-storage buffer system. The standard C file functions abstract the files from the application by using a dual-buffer system. The application uses a small buffer, for example a 34 byte string buffer. It reads from the filesystem, with for example the fgets function, 34-byte chunks. The filesystem reads data from the mass storage devices in 512 byte chunks (typical) or larger. It handles a caching buffer to store multiple 512 byte blocks. A typical FAT filesystem needs buffers for the FAT table, directory table and file data.

I.e. the filesystem reads data from mass storage device to data memory buffers (or cache) and the application reads data from this cache.
On a Dynamic C based system this involves copying several data blocks (FAT, directory, data) from the device to a single 512 byte root buffer, then copy this buffer to extended memory, then copy part of the extended memory buffer to a root data element.
Increasing the root buffer capacity will decrease the number of copies but we'll quickly end up with an out-of-root-data-memory error message.

The basic idea of the xgFAT libraries is to not use the intermediate buffer whenever possible. User interface file functions need to supply a direct memory address, either root or extended, to store the file data.

This results for example in a smoother looking GUI because screens are redraw faster and in faster webpage access as data is copied directly to the TCP/IP buffers

Caching the FAT and directory tables speeds up things but requires additional memory.
In systems which only access a few files, the speed increase by using no cache or only a few cached sectors will be not noticeable.

The actual caching system should be application dependant. The FS_CACHE.LIB includes several caching algorithms. Experiment with these to find the best trade off between memory usage and speed for your application.

2.4.3 Caching Algorithms

The FAT filesystem requires access to three types of sectors, the FAT table, the Directory entries, and the Data sectors. To reduce time-consuming re-reads of sectors an ‘embedded’ caching algorithm is used.

Embedded applications generally limit the use of mass storage data to:

- Reading/writing records from a database file
- Reading flat data: i.e. html, text, re-flash, graphic, audio data
- Writing flat data: i.e. recording audio files, dumping debug data

A huge cache buffer is useless for these applications. But if a single 1-sector buffer would be used for all FAT access, this buffer would need to re-read FAT and directory tables very often.

The xgFAT caching algorithm:

- Every device has a separate 2 sector (1024 bytes) FAT table buffer. One sector is used for reads, one for writes. If FAT_READ_ONLY is defined, only 1 sector per device is used. For FAT12 devices a 4 sector buffer is used. FAT12 always needs sector pairs due to the 1.5Byte length of each FAT entry.
- A cache memory pool is used to store all other read sectors. A single pool is used for all devices. The size of the pool can be adjusted with the FS_CACHESIZE macro. An intelligent algorithm is used to decide which sectors to purge when the pool is full.

- A dirty bit is used to mark sectors to be written. Writes are only done on user command or with a Cache Flush Timer (FAT_CACHE_TIMER macro).

The actual required size of the cache is application dependant. Applications which use a single device with occasional writes, will gain only a limited benefit from the cache. But applications with multiple attached devices and many random writes and file copies, will have a drastic increase in read/write speed by using the cache.

2.4.4 Battery-Backup SRAM

A battery-backup SRAM device does not use the cache. Reads and writes are always direct in the Rabbit memory map. No need for caching.

2.4.5 Flash

All Rabbit modules use an SST Flash chip with a hardware 4kByte sector size. To reduce flash write wear and increase the reading speed a specific cache buffer is used for the Flash device. This buffer can not be disabled with the DISABLE_FATCACHE macro.

Two 4kByte buffers are used in extended memory. The first Flash sector is copied in the first 4kByte buffer. This sector contains the BMP, the FAT table and the root directory. These 8 sectors are written to the flash on user request (FlushCache) or with the FAT Flush Timer. Care should be taken to flush the cache regularly, to prevent loss of FAT table changes. But also one should take care the cache is not written too often as this will reduce the Flash wear level (typical 100.000 writes).
Note that the flush cache command will only write to the Flash when something is changed in the first 8 sectors, i.e. after a file write or create.

The second 4kByte buffer is used for an 8-sector buffer. The flash will always be read or written in multiples of 8 sectors.

The xgFAT library uses a special algorithm to manage file writes to Flash. It will try to split up files in 4kByte chunks, i.e. a kind of a semi 4kByte cluster although the actual cluster size is only 1 sector. The algorithm will use the complete Flash for writes. It will not always write to the same empty sectors to get a better spread flash wear level.

2.4.6 xgFAT Memory Requirements

The xgFAT uses a minimum of program memory and still offers full FAT compatibility. Several features can be disabled (see 10.6) to even further reduce the program memory requirements.
Attaching multiple mass storage device does only slightly increase program memory requirements. Basically because an additional HAL is needed. The xgFAT libraries are written with function pointer algorithms to reduce program memory requirements.

Root memory requirements are restricted to basic data, needed to optimize FAT read/write speed.

- Each mounted volume requires a root memory struct. Depending on the maximum volume identifier length, this structures size is about 45 bytes.

Stack usage is limited. All functions use, whenever possible, local variables. The ‘auto’ keyword is not included in the variable declarations. But the default variable storage is ‘auto’ (on the stack) for Dynamic C 8.x and higher.

Depending on the chosen caching algorithm, the FAT cache uses extended and/or root ram memory (see 10.4). Note that there is absolutely no need to copy buffers to root memory. All FAT functions can operate directly on the data in the cache buffers with extended memory pointers.

2.5 xgFAT Macro’s

2.5.1 Device Selection

The xgFS_AutoMount.LIB library includes an initialization function which will attempt mounting FAT volumes on all attached mass storage devices. If this is not needed, a user initialization routine can be used or the following macro’s can be used.

The jumpstart macro’s:
XG_MMCD: enables MMC/SD device (default volume identifier S:\)
XG_CF: enables CF slot 1 device (default volume identifier C:\)
XG_CF2: enables CF slot 2 device (default volume identifier D:\)
XG_SF: enables Z-World Serial Flash device (default volume identifier Z:\)
XG_IDE: enables hard disc device (default volume identifier H:\)

BIOS macro’s:
XMEM_RESERVE_SIZE >= 0x10000: enables flash device (default volume identifier F:\)
FS2_RAM_RESERVE >= 16: enables bb-sram device (default volume identifier R:\)
(Important: don’t use the FS2.LIB if the FS2_RAM_RESERVE memory is used for the xgFAT.LIB)

Disable macro’s:
FS_SD_DISABLE: disable the MMC/SD device
FS_CF_DISABLE: disable the CF Slot 1 device
FS_CF2_DISABLE: disable the CF Slot 2 device
FS_SF_DISABLE: disable the Z-World Serial Flash device
FS_IDE_DISABLE: disable the hard disc device
FS_FLASH_DISABLE: disable the on-board Flash device
FS_BB_SRAM_DISABLE: disable the battery backup SRAM device

2.5.2 Memory Limits

MAXVOLUMES: maximum simultaneously opened volumes. This defaults to the number of enabled devices if xgFS_AutoMount.LIB is used. This macro only needs to be changed if extended partitions are used.

MAXFILES: maximum of simultaneous open files. Default is 2.

MAXDEVNAME: maximum length of the device name including the ‘:’. The minimum and default setting is 2.

ENABLE_FAT12_SUPPORT: Enables FAT12 support. By default FAT12 is NOT supported. FAT12 requires additional memory resources. Following the FAT standard, FAT12 support is only required for devices smaller then 2MByte. It can be used for devices up to 128MByte. Most off-the-shelf memory devices are pre-formatted with FAT16 or FAT32. If you want to be sure all devices can be used, enable FAT12 support. The on-board devices (Flash, bb-SRAM, Z-World SF) use an adapted version of FAT16. This reduces the memory footprint of the xgFAT library. The only disadvantage would be that these devices can not be read by an external PC. As these are all hardwired on the board, a PC can not read them anyhow.

DISABLE_FAT16_SUPPORT: Disable FAT16 support. Only use this macro if you're very sure you will never use media formatted with FAT16. By enabling this macro, on-board devices (Flash, bb-SRAM, Z-World SF) can not be used anymore. The memory reduction is limited if FAT32 is still enabled. If both FAT16 and FAT32 are disabled, there is a reduction in memory requirements.

DISABLE_FAT32_SUPPORT: Disable FAT32 support. Only use this macro if you're very sure you will never use media formatted with FAT32. The memory reduction is limited if FAT16 is still enabled. If both FAT16 and FAT32 are disabled, there is a reduction in memory requirements.

DISABLE_LFN: Disable long file name support. By enabling this macro, long file names will not be read anymore and can not be written anymore. Long file names already present on the attached devices will not be changed. To reduce memory usage, this is a reasonable option in embedded applications.

2.5.3 Cache

FS_CACHE_SIZE: Actual cache size in 512-byte sectors. The default is 8.

FAT_READ_ONLY: this macro removes the FAT write support. This drastically reduces memory usage. It's a useful option for systems which only need to read files from devices. Combined with DISABLE_FS_CACHE you get a minimum memory footprint.

DISABLE_FS_WRITE_CACHE: disables caching of FAT file system writes. All 'dirty' sectors will be written immediately.
Important: On-board Flash volumes are ALWAYS cached independent of the DISABLE_FS_WRITE_CACHE setting. Don't forget to flush the cache for on-board flash volumes to store any changes in the Flash chip.

DISABLE_FS_CACHE: Reduces the xgFAT file system cache to a minimum. A single FAT buffer is used for all volumes (1024 bytes for FAT16/32, 2048 bytes for FAT12). And a 1 sector data buffer is used for all volumes (512 bytes).
This system is useful in systems which only use a single volume and a limited number of simultaneous open files. Although this reduces ram requirements, it also limits read/write speed.
Write caching is NOT disabled with this option. If you want to use the DISABLE_FS_CACHE without write caching, you also need to define the DISABLE_FS_WRITE_CACHE macro.
By enabling the FAT_READ_ONLY option the FAT buffer ram requirements are further reduced (512 bytes for FAT16/32, 1024 bytes for FAT12).

FS_CACHE_TIMER: Default to 0. Any other value will enable a flush of the write cache every FS_CACHE_TIMER seconds. The fsTick() function must be called regularly.

2.6 xgFAT Functions

Tbf

Warranty

DELGEN warrants that the product delivered hereunder shall conform to the applicable DELGEN datasheet or mutually agreed upon specifications and shall be free from defects in material and workmanship under normal use and service for a period of 1 year from the applicable date of invoice. Products which are "samples", "design verification units", and/or "prototypes" are sold "AS IS," "WITH ALL FAULTS," and without a warranty.
If, during such warranty period, (1) DELGEN is notified promptly in writing upon discovery of any defect in the goods, including a detailed description of such defect; (2) such goods are returned to DELGEN, DDP DELGEN's facility accompanied by DELGEN's Returned Material Authorization form; and (3) DELGEN's examination of such goods discloses to DELGEN's satisfaction that such goods are defective and such defects are not caused by accident, abuse, misuse, neglect, alteration, improper installation, repair, improper testing, or use contrary to any instructions issued by DELGEN, DELGEN shall (at its sole option) either repair, replace, or credit Buyer the purchase price of such goods.
No goods may be returned to DELGEN without DELGEN's Returned Material Authorization form. Prior to any return of goods by Buyer pursuant to this Section, Buyer shall afford DELGEN the opportunity to inspect such goods at Buyer's location, and any such goods so inspected shall not be returned to DELGEN without its prior written consent.
DELGEN shall return any goods repaired or replaced under this warranty to Buyer transportation prepaid. The performance of this warranty does not extend the warranty period for any goods beyond that period applicable to the goods originally delivered.
The foregoing warranty constitutes DELGEN's exclusive liability, and the exclusive remedy of buyer, for any breach of any warranty or other nonconformity of the goods covered by this agreement. This warranty is exclusive, and in lieu of all other warranties, express, implied, or statutory, including without limitation any warranties of merchantability or fitness for a particular purpose. The sole and exclusive remedy for any breach of this warranty shall be as expressly provided herein.

Limitation on Liability

Notwithstanding anything to the contrary contained herein, DELGEN shall not, under any circumstances, be liable to Buyer or any third parties for consequential, incidental, indirect, exemplary, special, or other damages. DELGEN's total liability shall not exceed the total amount paid by Buyer to DELGEN hereunder. DELGEN shall not under any circumstances be liable for excess costs of reprocurement

Notice to Users

DELGEN PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS UNLESS A SPECIFIC WRITTEN AGREEMENT REGARDING SUCH INTENDED USE IS ENTERED INTO BETWEEN THE CUSTOMER AND DELGEN PRIOR TO USE.

Life-support devices or systems are devices or systems intended for surgical implantation into the body or to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs are always present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

All DELGEN products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. DELGEN products may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range. This strategy is believed to be more economical and effective. Additional testing or burn-in of an individual unit is available by special arrangement.

Software License Agreement

Notice to Users
This is a legal agreement between you (an individual or single entity, referred to hereinafter as "you") and DELGEN for the computer software product(s) including any accompanying explanatory written materials (the "Software").
BEFORE INSTALLING, COPYING OR OTHERWISE USING THE SOFTWARE, YOU MUST AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT.
If you agree, you are allowed to use the software. If you do not agree with the terms and conditions of this Agreement, you are not allowed to use the software and must destroy all copies of the software.

DELGEN licenses this software to its customers upon acceptance of all the terms and conditions of this license agreement. Please read the terms carefully before downloading or installing the software.
If you do not accept all the terms, you may not install or use this software, and should contact your sales representative to receive a full refund.
If you have any questions, call +32-475-60.64.33, or write to the DELGEN office at 241, route de Longwy, LU-1941 Luxembourg, GD-Luxembourg.

1. Definitions. "Software" means the accompanying computer programs, data compilation(s), and documentation. "You" means the licensee, and are referred to as "You."

2. Term. The term of the license granted herein shall continue until terminated either (a) by You, for your convenience, by written notice to DELGEN or (b) automatically if a material breach by You is not cured within thirty (30) days of such breach. Immediately upon any termination of this license for any reason, You must return to DELGEN all copies of the Software.

3. License Grant. You are granted non-exclusive rights to install and use the Software on a single computer only; however, if the Software is permanently installed on the hard disk or other storage device of a computer (other than a network server), and one person uses that computer more than 80% of the time, then that person may also use the Software on a portable or home computer. You may not install the Software on a network or transmit the Software electronically from one computer to another or over a network. You may copy the Software for archival purposes, provided that any copy must contain the original Software's proprietary notices in unaltered form.

4. Restrictions. You may not: (i) rent, lease, sublicense, loan, timeshare, or permit others to use the Software, except as expressly provided above; (ii) modify or translate the Software; (iii) reverse engineer, decompile, or disassemble the Software, except to the extent this restriction is expressly prohibited by applicable law; (iv) except as permitted by Section 5 below, create a derivative work based on the Software or merge the Software with another product; (vi) copy the Software, except that a reasonable number of copies may be made for archival purposes; or (vii) remove or obscure any proprietary rights notices or labels.

5. Transfers. You may not transfer or assign, in any manner, including by operation of law, the Software or any rights under this Agreement without the prior written consent of DELGEN, which consent shall not be unreasonably withheld. A condition to any transfer or assignment shall be that the recipient agrees to the terms of this Agreement. Any attempted transfer or assignment in violation of this provision shall be null and void.

6. Ownership. DELGEN and its suppliers own the Software and all intellectual property rights embodied therein, including patents, copyrights and valuable trade secrets embodied in the Software's design and coding methodology. The Software is protected by

Change List

1.0

Initial release